Feasibly Reducing KAT Equations to KA Equations

James Worthington
Mathematics Department, Cornell University
Ithaca, NY 14853-4201 USA
worthing@math.cornell.edu

February 2, 2008

Abstract

Kleene algebra (KA) is the algebra of regular events. Familiar examples of Kleene algebras include regular sets, relational algebras, and trace algebras. A Kleene algebra with tests (KAT) is a Kleene algebra with an embedded Boolean subalgebra. The addition of tests allows one to encode `while` programs as KAT terms, thus the equational theory of KAT can express (propositional) program equivalence. More complicated statements about programs can be expressed in the Hoare theory of KAT, which suffices to encode Propositional Hoare Logic.

That the equational theory of KAT reduces to the equational theory of KA has been shown by Cohen et al. Unfortunately, their reduction involves an exponential blowup in the size of the terms involved. Here we give an alternate feasible reduction.

1 Introduction

The class of Kleene algebras is defined by equations and equational implications over the signature $\{0, 1, +, \cdot, ^*\}$. Some well-known examples of Kleene algebras include relational algebras, trace algebras, and sets of regular languages (see [1] for more examples and applications). In fact, the set of regular languages over an alphabet Σ is the free Kleene algebra on Σ. That is, given two KA terms α and β, $\alpha = \beta$ modulo the axioms of Kleene algebra if and only if α and β denote the same regular set [4]. A Kleene algebra with tests is a Kleene algebra with an embedded Boolean subalgebra (the complementation function is only defined on Boolean terms).

Adding tests allows the encoding of `while` programs as KAT terms. As a result, the equational theory of KAT suffices to express (propositional) equivalence of `while` programs. Moreover, Propositional Hoare Logic can be encoded in the Hoare theory of KAT (equational implications of the form $r = 0 \to p = q$), and furthermore the Hoare theory of KAT reduces efficiently to the equational theory of KAT. Combining all of these reductions shows that the equational theory of KA can be used to express interesting properties of programs succinctly. See [6], [8], and [9] for details.

In [5], it is shown that the equational theory of KAT reduces to the equational theory of KA. Unfortunately, the reduction used can increase the size of the terms involved exponentially. We given alternate reduction, which increases the size of the terms by only a polynomial amount. This paper is organized as follows. In section 2, we provide the relevant definitions and recall the encoding of finite automata as Kleene algebra terms. In section 3, we prove some useful theorems of Kleene algebra used for reasoning about automata and give an overview of *guarded string algebras*. In section 4, we give a feasible reduction from a KAT term to an automaton encoded as a KA term. In section 5, we remark that the Hoare theory of KA(T) can be efficiently reduced to the equational theory of KA(T), and in section 6 we make an observation concerning automata constructed from KAT terms representing deterministic `while` programs.

2 Background

In this section, we describe our proof system and recall some useful facts about KA(T). The axiomatization of Kleene algebra, results about matrices, and the encoding of automata as KA terms are from [4]. The definition of KAT is from [6].

2.1 Equational Logic

By "proof", we mean a sequent in the equational implication calculus. Let $\alpha, \beta, \gamma, \delta$ be terms in the language of Kleene algebra. The equational axioms are:

$$\begin{array}{l}
\alpha = \alpha \\
\alpha = \beta \rightarrow \beta = \alpha \\
\alpha = \beta \rightarrow \beta = \gamma \rightarrow \alpha = \gamma \\
\alpha = \beta \rightarrow \gamma = \delta \rightarrow \alpha + \gamma = \beta + \delta \\
\alpha = \beta \rightarrow \gamma = \delta \rightarrow \alpha \cdot \gamma = \beta \cdot \delta \\
\alpha = \beta \rightarrow \alpha^* = \beta^*.
\end{array}$$

We consider these Horn formulas to be implicitly universally quantified.

Let Φ be a sequence of equations or equational implications, e an equation, ϕ a Horn formula, and ψ an equational axiom or an axiom of KA (given below). Let σ be a substitution of terms for variables. The rules of inference are:

$$\vdash \sigma(\psi) \qquad e \vdash e \qquad \frac{\Phi \vdash \phi}{\Phi, e \vdash \phi} \qquad \frac{\Phi, e \vdash \phi}{\Phi \vdash e \rightarrow \phi} \qquad \frac{\Phi \vdash e \qquad \Phi \vdash e \rightarrow \phi}{\Phi \vdash \phi},$$

and the structural rules which allow us to treat a sequence of formulas as a set of formulas. For a proof that this is a complete deductive system, see [10]. We also allow "substitution of equals for equals". For example, from $a = b$, conclude $c(a + 1) = c(b + 1)$ in one step.

2.2 Kleene Algebra

We now state the axioms of Kleene algebra. The first are the idempotent semiring axioms. Note that we abbreviate $\alpha \cdot \beta$ as $\alpha\beta$.

1. $(a + b) + c = a + (b + c)$
2. $a + b = b + a$
3. $a + 0 = a$
4. $a + a = a$
5. $(ab)c = a(bc)$
6. $1a = a1 = a$
7. $a(b + c) = ab + ac$
8. $(a + b)c = ac + bc$
9. $0a = 0a = 0$

In any idempotent semiring, addition can be used to define a partial order:

$$x \leq y \Leftrightarrow x + y = y.$$

For brevity, we add the symbol $\leq$ to the language.

There are four axioms involving *. The equational axioms are:

10. $1 + xx^* = x^*$
11. $1 + x^*x = x^*$

There are also two equational implications:

12. $b + ax \leq x \rightarrow a^*b \leq x$
13. $b + xa \leq x \rightarrow ba^* \leq x$

The equational implications guarantee unique least solutions to the linear inequalities

$$b + aX \leq X$$

$$b + Xa \leq X$$

in the presence of the other axioms.

2.3 Kleene Algebra with Tests

A Kleene algebra with tests is a Kleene algebra with an embedded Boolean subalgebra; Boolean terms are called tests. Formally, a Kleene algebra with tests is a two-sorted structure $(K, B, +, \cdot, ^*, ^-, 0, 1)$ such that $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra and $(B, +, \cdot, ^-, 0, 1)$ is a Boolean algebra. Note that complementation is only defined on tests.

We use the following axiomatization of Boolean algebra. Let b, c, d be Boolean terms.

1. KA axioms 1 - 9
2. $\overline{0} = 1$; $\overline{1} = 0$
3. $b + 1 = 1$
4. $b\overline{b} = \overline{b}b = 0$
5. $\overline{\overline{b}} = b$
6. $bb = b$
7. $\overline{b + c} = \overline{b}\overline{c}$; $\overline{bc} = \overline{b} + \overline{c}$
8. $bc = cb$
9. $b + cd = (b + c)(b + d)$

Any Boolean term b satisfies $b \leq 1$. Since $1^* = 1$ and * is monotonic, the KA axioms imply $b^* = 1$. Note that any Kleene algebra can be viewed as a KAT with $\{0, 1\}$ as the two-element Boolean subalgebra.

2.4 Matrices and Automata

The Kleene algebra axioms imply that the set of $n \times n$ matrices over a KA also forms a KA. Addition and multiplication of matrices are defined in the usual way, 0 is interpreted as the $n \times n$ zero matrix, and 1 as I_n. Equality and the partial order $\leq$ are defined componentwise. To define the star of an $n \times n$ matrix, we first define the star of a 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a + bd^*c)^* & (a + bd^*c)bd^* \\ (d + ca^*b)ca^* & (d + ca^*b)^* \end{bmatrix}.$$

We then extend this definition to arbitrary square matrices inductively. Given a square matrix E, partition E into four submatrices

$$E = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

such that A and D are square. By induction, A^* and D^* exist. Let $F = A + BD^*C$. Then

$$E^* = \left[\begin{array}{c|c} F^* & F^*BD^* \\ \hline D^*CF^* & D^* + D^*CF^*BD^* \end{array}\right].$$

It is a consequence of the KA axioms that any partition may be chosen to compute E^*.

In [3], it is shown that the set of $n \times n$ matrices over a Kleene algebra with tests is a Kleene algebra with tests. The Boolean subalgebra is the set of matrices with Boolean terms on the diagonal and all other entries equal to 0.

At several points in the proof below, we will have to reason about non-square matrices. We would like to know whether the theorems of Kleene algebra hold when the primitive letters are interpreted as matrices of arbitrary dimension and the function symbols are treated polymorphically. In general, the answer is no. However, there is a large class of theorems for which this does hold, and they suffice for our purposes. See [7] for a thorough treatment of this issue.

We now recall how to use matrices over a KA to encode finite automata.

Definition 1. *An* automaton *over a Kleene algebra K is a triple (u, A, v) where u and v are n-dimensional vectors with entries from $\{0, 1\}$ and A is an $n \times n$ matrix over K. The vector u encodes the start states of the automaton and is called the* start vector. *The vector v encodes the accept states of the automaton and is called the* accept vector. *The matrix A is called the* transition matrix. *The language accepted by (u, A, v) is $u^{\mathrm{T}}A^*v$. The* size *of (u, A, v) is the number of states, i.e., if A is an $n \times n$ matrix, then the size of (u, A, v) is n.*

This definition is a bit general for the purposes at hand. Given an alphabet Σ, let $\mathcal{F}_\Sigma$ be the free Kleene algebra on generators Σ. Over $\mathcal{F}_\Sigma$, the definition of an automaton given above is essentially the same as the classical definition of a finite automaton. In the sequel, all automata are over some $\mathcal{F}_\Sigma$. Furthermore, most of the automata we consider have uncomplicated transition matrices.

Definition 2. *Let (u, A, v) be an automaton over $\mathcal{F}_\Sigma$. The automaton (u, A, v) is* simple *if A can be expressed as a sum*

$$A = J + \sum_{a \in \Sigma} a \cdot A_a$$

where J and each A_a is a 0-1 matrix.
The automaton (u, A, v) is ϵ-free *if J is the zero matrix.*
The automaton (u, A, v) is deterministic *if it is simple, ϵ-free, and u and all rows of each A_a have exactly one 1.*

Given an automaton (u, A, v), we denote the transition relation encoded by A as δ_A, and the extended transition relation defined on (states,words) as $\hat{\delta}_A$. Given an $a \in \Sigma$, we denote the restriction of δ_A to only a-transitions by δ_A^a. For transition matrices A, B, C, we denote the underlying state sets of the automata by $\mathcal{A}, \mathcal{B}, \mathcal{C}$. We now state the theorems of KA which we will use to reason about automata.

3 Useful Theorems of KA

The completeness result of [4] uses the fact that automata can be encoded as KA terms. To simplify proofs, we add several theorems of Kleene algebra involving automata to our list of allowable rules of inference. For each theorem we add, it will be clear that the hypotheses of the theorem are easy to check, so proofs constructed using these new rules of inference are verifiable in polynomial time. Several of the theorems about automata are based on the following theorems of Kleene algebra:

$$(x + y)^* = x^*(yx^*)^*$$

$$ay = yb \rightarrow a^*y = yb^*$$

$$x(yx)^* = (xy)^*x.$$

These are known as the *denesting, bisimulation*, and *sliding* rules, respectively. See [4] for a proof that these rules are consequences of the KA axioms.

We now provide an overview of *guarded string algebras*, which are models of the KAT axioms. For a more detailed introduction, see [5]. Guarded string algebras play the same role for KAT that regular languages do for KA; two KAT terms t_1 and t_2 are equivalent modulo the axioms of Kleene algebra with tests if and only if they denote the same set of guarded strings.

Let P and B be finite alphabets. Elements of P are called atomic programs, and elements of B are called primitive tests (to distinguish them from atomic elements of the Boolean algebra generated by B). Guarded strings are obtained from each word $w \in P^*$ by interspersing atoms of the free Boolean algebra on B among the letters of w (we require that a guarded string both begins and ends with an atom). Let $b_1, b_2, ..., b_n$ be the elements of B. Recall that an atom α of the free Boolean algebra on B is a product of the form

$$\alpha = c_1c_2 \cdots c_n$$

where $c_i \in \{b_i, \overline{b_i}\}$ for each i. We require an ordering on the literals appearing in an atom so that there is a unique string denoting each atom. Let A_B denote the set of atoms.

Given a guarded string x, let first(x) be the leftmost atom of x, and last(x) be the rightmost atom of x. We define a partial concatenation operation on guarded strings, denoted $\diamond$, as follows. Given two guarded strings, x and y, let $x = x'\alpha$ and $y = \beta y'$, where $\alpha =$last(x) and $\beta =$ first(y).

Define

$$x \diamond y = x'\alpha y', \text{ if } \alpha = \beta, \text{ undefined otherwise.}$$

We now give interpretations of the KAT operations on sets of guarded strings. Let C and D be sets of guarded strings. Define

$$\begin{aligned} &C + D = C \cup D \\ &C \cdot D = \{x \diamond y \mid x \in C,\ y \in D\} \\ &C^0 = A_B \\ &C^* = \textstyle\bigcup_{n \geq 0}\ C^n. \end{aligned}$$

We must also interpret the complementation function. Let C be a set of guarded strings such that $C \subseteq A_B$. Define

$$\overline{C} = A_B - C.$$

Using these operations, we can define a function G from KAT terms to sets of guarded strings inductively. The base cases are:

$$\begin{aligned}
&G(0) = \emptyset \\
&G(1) = \{\alpha \mid \alpha \in A_B\} \\
&G(b) = \{\alpha \mid \alpha \to b \text{ is a propositional tautology}\} \\
&G(p) = \{\alpha p \beta \mid \alpha, \beta \in A_B\}.
\end{aligned}$$

In [5], the completeness of the guarded string model for the equational theory of KAT is shown by a reduction from the equational theory of KAT to the equational theory of KA. This is achieved by transforming a KAT term t into a KAT-equivalent term t' such that $R(t') = G(t)$. Unfortunately, the term t' may be exponentially longer than t. We give an alternate construction. Given a term t, we construct an automaton (u, A, v) such that $t = u^{\mathrm{T}} A^* v$ modulo the axioms of KAT, and (u, A, v) accepts precisely the set of guarded strings denoted by t. The automaton (u, A, v) will be polynomial in the size of t.

We need a few additional theorems of Kleene algebra in our construction. The extra axioms satisfied by Boolean terms, particularly multiplicative idempotence and star-triviality, complicate the construction of the automaton. We overcome these difficulties by selectively applying the Boolean axioms to Boolean terms. That is, we first treat Boolean terms simply as words over an alphabet, and apply the lemmas below. However, these lemmas produce automata which are not simple. In the inductive construction in section 4.3 we then use the Boolean axioms to simplify the transition matrices. Note, however, that the two lemmas below are theorems of Kleene algebra, and do not require the Boolean axioms.

3.1 The KAT Concatenation Lemma

The *KAT concatenation* lemma is based on the following alternate way of constructing an automaton accepting the concatenation of two languages. The standard construction of such an automaton is to connect the accept states of the first automaton to the start states of the second with ϵ-transitions. However, we could also do the following: for each state i of (u, A, v) with an outgoing x transition to an accept state, and each state j of (s, B, t) with an incoming y transition from a start state, add an xy transition from i to j. Note that we allow x and y to be arbitrary elements of a Kleene algebra, not just letters in Σ. This construction yields an automaton accepting $u^{\mathrm{T}} A^* v s^{\mathrm{T}} B^* t$, provided neither (u, A, v) nor (s, B, t) has a state which is both a start state and an accept state, which we can represent algebraically as $u^{\mathrm{T}} v = 0,\ s^{\mathrm{T}} t = 0$. This idea is the crux of the KAT concatenation lemma. The lemma itself looks rather complicated, so we explain how it will be used. In the construction in 5.2, we will have two ϵ-free automata, (u_1, A_1, v_1) and (u_2, A_2, v_2). Each of these automata will be the disjoint union of two automata:

$$(u_i, A_i, v_i) = \left(\left[\begin{array}{c} o_i \\ \hline s_i \end{array} \right], \left[\begin{array}{c|c} C_i & 0 \\ \hline 0 & B_i \end{array} \right], \left[\begin{array}{c} r_i \\ \hline t_i \end{array} \right] \right).$$

It will be the case that neither of them accept the empty word, i.e.,

$$o_i^{\mathrm{T}} r_i = 0$$

$$s_i^{\mathrm{T}} t_i = 0$$

for $i = 1, 2$. The construction will require an automaton accepting

$$L = (o_1^{\mathrm{T}} C_1^* r_1 s_2^{\mathrm{T}} B_2^* t_2) + (s_1^{\mathrm{T}} B_1^* t_1 o_2^{\mathrm{T}} C_2^* r_2) + (s_1^{\mathrm{T}} B_1^* t_1 s_2^{\mathrm{T}} B_2^* t_2).$$

Let Φ be a sequence of equations or equational implications. The KAT concatenation lemma,

$$\frac{\Phi \vdash o_1^{\mathrm{T}} r_1 = 0 \qquad \Phi \vdash o_2^{\mathrm{T}} r_2 = 0 \qquad \Phi \vdash s_1^{\mathrm{T}} t_1 = 0 \qquad \Phi \vdash s_2^{\mathrm{T}} t_2 = 0}{\Phi \vdash \begin{bmatrix} o_1 \\ s_1 \\ 0 \\ 0 \end{bmatrix}^{\mathrm{T}} \left[\begin{array}{cc|cc} C_1 & 0 & 0 & C_1 r_1 s_2^{\mathrm{T}} B_2 \\ 0 & B_1 & B_1 t_1 o_2^{\mathrm{T}} C_2 & B_1 t_1 s_2^{\mathrm{T}} B_2 \\ \hline 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & B_2 \end{array}\right]^* \begin{bmatrix} 0 \\ 0 \\ r_2 \\ t_2 \end{bmatrix} = L}$$

allows us to do this.

The proof is a straightforward calculation:

$$\begin{bmatrix} o_1 \\ s_1 \\ 0 \\ 0 \end{bmatrix}^{\mathrm{T}} \left[\begin{array}{cc|cc} C_1 & 0 & 0 & C_1 r_1 s_2^{\mathrm{T}} B_2 \\ 0 & B_1 & B_1 t_1 o_2^{\mathrm{T}} C_2 & B_1 t_1 s_2^{\mathrm{T}} B_2 \\ \hline 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & B_2 \end{array}\right]^* \begin{bmatrix} 0 \\ 0 \\ r_2 \\ t_2 \end{bmatrix} =$$

$$o_1^{\mathrm{T}} C_1^* C_1 r_1 s_2^{\mathrm{T}} B_2 B_2^* t_2 + s_1^{\mathrm{T}} B_1^* B_1 t_1 o_2^{\mathrm{T}} C_2 C_2^* r_2 + s_1^{\mathrm{T}} B_1^* B_1 t_1 s_2^{\mathrm{T}} B_2 B_2^* t_2.$$

Using the hypotheses, it is easy to show that this sum is equal to L. The proofs involved are of the following form:

$$\begin{aligned} o_1^{\mathrm{T}} C_1^* r_1 &= o_1^{\mathrm{T}} (1 + C_1^* C_1) r_1 \\ &= o_1^{\mathrm{T}} r_1 + o_1^{\mathrm{T}} C_1^* C_1 r_1 \\ &= o_1^{\mathrm{T}} C_1^* C_1 r_1. \end{aligned}$$

3.2 The KAT Asterate Lemma

Let (u, A, v) be a simple, ϵ-free automaton and γ be a regular expression. Suppose $u^{\mathrm{T}} A^* v = \gamma$. The standard construction of an automaton accepting $\gamma\gamma^*$ proceeds by adding ϵ-transitions from the accept states of (u, A, v) back to its start states. Suppose (u, A, v) has no paths of length 0 or 1 from a start state to an accept state, which we can model algebraically as $u^{\mathrm{T}} v = 0, u^{\mathrm{T}} A v = 0$. In this case, we can construct an automaton accepting $\gamma\gamma^*$ from (u, A, v) with the following procedure: for each state i with an outgoing x transition to an accept state, and each state j with an incoming y transition from a start state, add an xy transition from i to j. This automaton, although not simple, accepts $\gamma\gamma^*$. This idea is the basis of the *KAT asterate* lemma.

Suppose (u, A, v) is the disjoint union of two automata, (o, C, r) and (s, B, t). Also suppose that $o^{\mathrm{T}} C^* r \leq 1$, and $s^{\mathrm{T}} t + s^{\mathrm{T}} B t = 0$, which implies $s^{\mathrm{T}} B^* t = s^{\mathrm{T}} B^* B B t$. Under these conditions, we can apply the KAT asterate lemma:

$$\frac{\Phi \vdash o^{\mathrm{T}} C^* r \leq 1 \qquad \Phi \vdash s^{\mathrm{T}} B^* t = s^{\mathrm{T}} B^* B B t}{\Phi \vdash \left(\left[\begin{array}{c} o \\ \hline s \end{array}\right]^{\mathrm{T}} \left[\begin{array}{c|c} C & 0 \\ \hline 0 & B \end{array}\right]^* \left[\begin{array}{c} r \\ \hline t \end{array}\right] \right)^* = \left[\begin{array}{c} 1 \\ \hline s \end{array}\right]^{\mathrm{T}} \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & B + B t s^{\mathrm{T}} B \end{array}\right]^* \left[\begin{array}{c} 1 \\ \hline t \end{array}\right]}.$$

Note that $B + Bts^{\mathrm{T}} B$ algebraically encodes the alternate asterate construction.

Since (u, A, v) is the disjoint union of (o, C, r) and (s, B, t), it is easy to show that

$$u^{\mathrm{T}} A^* v = o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t.$$

By KA axiom 10,

$$(u^{\mathrm{T}} A^* v)^* = 1 + u^{\mathrm{T}} A^* v (u^{\mathrm{T}} A^* v)^*.$$

We can now substitute:

$$1 + u^{\mathrm{T}} A^* v (u^{\mathrm{T}} A^* v)^* = 1 + (o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)(o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)^*.$$

By the denesting rule of Kleene algebra,

$$1 + (o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)(o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)^* = 1 + (o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)(o^{\mathrm{T}} C^* r)^* (s^{\mathrm{T}} B^* t (o^{\mathrm{T}} C^* r)^*)^*.$$

Since $o^{\mathrm{T}} C^* r \leq 1, (o^{\mathrm{T}} C^* r)^* = 1$. We can simplify:

$$1 + (o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)(o^{\mathrm{T}} C^* r)^* (s^{\mathrm{T}} B^* t (o^{\mathrm{T}} C^* r)^*)^* = 1 + (o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)(s^{\mathrm{T}} B^* t)^*.$$

By distributivity and axiom 10 again,

$$1 + (o^{\mathrm{T}} C^* r + s^{\mathrm{T}} B^* t)(s^{\mathrm{T}} B^* t)^* = 1 + s^{\mathrm{T}} B^* t (s^{\mathrm{T}} B^* t)^*.$$

At this point, we have shown that $u^{\mathrm{T}} A^* v = 1 + s^{\mathrm{T}} B^* t (s^{\mathrm{T}} B^* t)^*$. It remains to be shown that under the assumption $s^{\mathrm{T}} B^* t = s^{\mathrm{T}} B^* B B t$,

$$s^{\mathrm{T}} B^* t (s^{\mathrm{T}} B^* t)^* = s^{\mathrm{T}} (B + B t s^{\mathrm{T}} B)^* t. \qquad (1)$$

Reasoning algebraically,

$$\begin{aligned} s^{\mathrm{T}} B^* t (s^{\mathrm{T}} B^* t)^* &= s^{\mathrm{T}} B^* B B t (s^{\mathrm{T}} B^* B B t)^* \\ &= s^{\mathrm{T}} B^* B (B t s^{\mathrm{T}} B^* B)^* B t \\ &= s^{\mathrm{T}} B B^* (B t s^{\mathrm{T}} B B^*)^* B t \\ &= s^{\mathrm{T}} B (B + B t s^{\mathrm{T}} B)^* B t. \end{aligned}$$

The following equation is an easy consequence of the axioms of Kleene algebra:

$$(B + B t s^{\mathrm{T}} B)^* = 1 + B t s^{\mathrm{T}} B (B + B t s^{\mathrm{T}} B)^* + (B + B t s^{\mathrm{T}} B)^* B t s^{\mathrm{T}} B + B (B + B t s^{\mathrm{T}} B)^* B.$$

Multiplying the equation on the left by s^{T}, on the right by t, and simplifying using $s^{\mathrm{T}} t = 0$ and $s^{\mathrm{T}} B t = 0$ yields

$$s^{\mathrm{T}} (B + B t s^{\mathrm{T}} B)^* t = s^{\mathrm{T}} B (B + B t s^{\mathrm{T}} B)^* B t.$$

This proves (1). We now add the trivial one-state automaton to the automaton $(s, B + B t s^{\mathrm{T}} B, t)$, completing the proof of the KAT asterate lemma.

4 KAT Term to Automaton

In this section, we give the transducer which takes as input a KAT term t and outputs an automaton accepting $G(t)$. Before constructing the automaton, it must convert t into a well-behaved form.

4.1 Only Complement Primitive Tests

The machine first uses the De Morgan laws and the Boolean axiom $\overline{\overline{b}} = b$ to transform a term t into an equivalent term t' in which the complementation symbol is only applied to atomic tests. If we interpret t' as a regular expression, then $R(t') \subseteq (P \cup B \cup \overline{B})^*$, where $\overline{B} = \{\overline{b} \mid b \in B\}$. The transducer works as follows. On input t, it copies t onto its worktape and onto the output tape. Then, starting at the root of the syntax tree of t, it works it way down the tree until it finds a subtree containing only Boolean terms such that either some term is complemented twice, or a conjunction or disjunction appears under the complement symbol. It then applies the appropriate axiom to this subtree, overwrites its worktape contents, and then outputs the updated term. The machine then begins searching again at the root of the tree. When it scans the whole tree and does not have to apply any axioms, it stops. The transducer requires only polynomially many worktape cells. Furthermore, the increase in the size of the term is negligible. At the end of this stage, it has t' written on its worktape.

4.2 New variables for atoms

For the remainder of the construction, it is advantageous to treat each atom as a single letter. Let $z = 2^{|B|}$. The machine generates z many new variables, $x_1, x_2, ..., x_z$. For each i, it outputs the equation

$$x_i = \alpha_i$$

where α_i is the i^{th} atom. The automaton constructed below uses the alphabet $P \cup \{x_1, x_2, ..., x_z\}$. It is a routine matter to verify that two KAT terms denote same set of guarded strings if and only if they denote the same set of words after performing this substitution. For the rest of the construction, we use the terms "guarded strings" and "guarded strings after this substitution" interchangeably.

4.3 Constructing the Automaton

Now that the preprocessing of the term is complete, the machine constructs the automaton. The construction is inductive and resembles the construction the proof of Kleene's theorem. However, the machine will maintain several invariants throughout the construction which were not necessary in the pure Kleene algebra case. At a given substage, let (u, A, v) be the final automaton constructed. The automaton (u, A, v) will satisfy:

- (u, A, v) is simple and ϵ-free.
- (u, A, v) is the "disjoint union" of two automata, (o, C, r) and (s, B, t), or just (o, C, r), or just (s, B, t).
- (s, B, t) accepts only words of length two or more, so., $s^{\mathrm{T}} B^* t = s^{\mathrm{T}} B^* B B t$.
- (o, C, r) is a two state automaton accepting only one-letter words from the alphabet $\{x_1, x_2, ..., x_z\}$.
- The first two states of (u, A, v) are the states of (o, C, r) (if (o, C, r) is nonempty).

The base case of the induction is as follows. For an atomic term a, $\hat{a}$ denotes the automaton constructed. For an atom x_i and a primitive test b, $x_i \leq b$ means that $x_i \to b$ is a propositional tautology.

$$\hat{0} = (0, 0, 0)$$

$$\hat{1} = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & \sum_i x_i \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)$$

$$\hat{b} = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & \sum_{x_i \leq b} x_i \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)$$

$$\hat{p} = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & \sum_i x_i & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & \sum_i x_i \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right)$$

For each automaton, the machine must prove that the language it accepts is KAT-equivalent to the appropriate atomic term. There are finitely many atomic terms, so the machine can store all of the necessary proofs in its finite control. Note that this expansion increases the size of a term by only a constant amount, although the constant is exponential in $|B|$. Cf. the proof that the Boolean algebra axioms entail all propositional tautologies.

We now treat the inductive step of the construction. The easiest automaton to construct is that for addition. Suppose we have two automata (u_1, A_1, v_1) and (u_2, A_2, v_2), such that $u_1^{\mathrm{T}} A_1^* v_1 = \gamma$ and $u_2^{\mathrm{T}} A_2^* v_2 = \delta$. By induction, (u_1, A_1, v_1) is the disjoint union of (o_1, C_1, r_1) and (s_1, B_1, t_1), and (u_2, A_2, v_2) is the disjoint union of (o_2, C_2, r_2) and (s_2, B_2, t_2). The machine first proves the equations

$$\begin{aligned} u_1^{\mathrm{T}} A_1^* v_1 &= o_1^{\mathrm{T}} C_1^* r_1 + s_1^{\mathrm{T}} B_1^* t_1 \\ u_2^{\mathrm{T}} A_2^* v_2 &= o_2^{\mathrm{T}} C_2^* r_2 + s_2^{\mathrm{T}} B_2^* t_2. \end{aligned}$$

It then outputs a proof that

$$\gamma + \delta = (o_1^{\mathrm{T}} C_1^* r_1 + o_2^{\mathrm{T}} C_2^* r_2) + s_1^{\mathrm{T}} B_1^* t_1 + s_2^{\mathrm{T}} B_2^* t_2.$$

The machine can now construct a two-state automaton (o, C, r) which accepts $(o_1^{\mathrm{T}} C_1^* r_1 + o_2^{\mathrm{T}} C_2^* r_2)$, then apply the addition construction from 4.1 to $(o, C, r), (s_1, B_1, t_1)$, and (s_2, B_2, t_2). This yields an automaton (u, A, v) which satisfies the invariants and accepts $\gamma + \delta$. Note that there are only finitely many possibilities for (o_1, C_1, r_1) and (o_2, C_2, r_2), so the machine can prove

$$o^{\mathrm{T}} C^* r = o_1^{\mathrm{T}} C_1^* r_1 + o_2^{\mathrm{T}} C_2^* r_2$$

using data from its finite control.

The automaton for the product of two terms is more complicated. Again, let (u_1, A_1, v_1) and (u_2, A_2, v_2) be two automata such that $u_1^{\mathrm{T}} A_1^* v_1 = \gamma$ and $u_2^{\mathrm{T}} A_2^* v_2 = \delta$. As in the case for addition, we use the fact that each of these automata is the disjoint union of two automata:

$$\begin{aligned} u_1^{\mathrm{T}} A_1^* v_1 &= o_1^{\mathrm{T}} C_1^* r_1 + s_1^{\mathrm{T}} B_1^* t_1 \\ u_2^{\mathrm{T}} A_2^* v_2 &= o_2^{\mathrm{T}} C_2^* r_2 + s_2^{\mathrm{T}} B_2^* t_2. \end{aligned}$$

The machine can output a proof of the equations

$$\begin{aligned} \gamma\delta &= (o_1^{\mathrm{T}} C_1^* r_1 + s_1^{\mathrm{T}} B_1^* t_1)(o_2^{\mathrm{T}} C_2^* r_2 + s_2^{\mathrm{T}} B_2^* t_2) \\ &= (o_1^{\mathrm{T}} C_1^* r_1 o_2^{\mathrm{T}} C_2^* r_2) + (o_1^{\mathrm{T}} C_1^* r_1 s_2^{\mathrm{T}} B_2^* t_2) + (s_1^{\mathrm{T}} B_1^* t_1 o_2^{\mathrm{T}} C_2^* r_2) + (s_1^{\mathrm{T}} B_1^* t_1 s_2^{\mathrm{T}} B_2^* t_2). \end{aligned}$$

The term $(o_1^{\mathrm{T}} C_1^* r_1 o_2^{\mathrm{T}} C_2^* r_2)$ is a sum of atoms after simplifying using the Boolean axioms. The machine can construct a two-state automaton (o, C, r) accepting this sum. Since there are only finitely many choices for $o_1^{\mathrm{T}} C_1^* r_1$ and $o_2^{\mathrm{T}} C_2^* r_2$, all of the necessary proofs can be stored in the finite control of the machine.

Let (s, B, t) be the automaton

$$\left(\begin{bmatrix} o_1 \\ s_1 \\ 0 \\ 0 \end{bmatrix}, \left[\begin{array}{cc|cc} C_1 & 0 & 0 & C_1 r_1 s_2^{\mathrm{T}} B_2 \\ 0 & B_1 & B_1 t_1 o_2^{\mathrm{T}} C_2 & B_1 t_1 s_2^{\mathrm{T}} B_2 \\ \hline 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & B_2 \end{array} \right], \begin{bmatrix} 0 \\ 0 \\ r_2 \\ t_2 \end{bmatrix} \right).$$

The machine first outputs proofs of the hypotheses of the KAT concatenation lemma. It can then output

$$s^{\mathrm{T}} B^* t = (o_1^{\mathrm{T}} C_1^* r_1 s_2^{\mathrm{T}} B_2^* t_2) + (s_1^{\mathrm{T}} B_1^* t_1 o_2^{\mathrm{T}} C_2^* r_2) + (s_1^{\mathrm{T}} B_1^* t_1 s_2^{\mathrm{T}} B_2^* t_2),$$

which follows from the KAT concatenation lemma.

The machine now constructs a simple automaton (s, B', t) by simplifying the transition matrix for (s, B, t) using the Boolean axioms and outputs a proof of the equivalence of (s, B, t) and (s, B', t). It then adds the automata (o, C, r) and (s, B', t) together to get (u, A, v), and outputs a proof of the equation

$$u^{\mathrm{T}} A^* v = \gamma\delta.$$

Finally, we come to the construction for *. Let (u, A, v) be an automaton such that $u^{\mathrm{T}} A^* v = \gamma$. This automaton is the disjoint union of two automata, (o, C, r) and (s, B, t) such that (o, C, r) accepts a sum of atoms and (s, B, t) accepts no words of length less than two. The machine first outputs proofs that

$$o^{\mathrm{T}} C^* r \leq 1$$

$$s^{\mathrm{T}} B^* B B t = s^{\mathrm{T}} B t.$$

These facts follow from the Boolean axioms and the equation $s^{\mathrm{T}} t + s^{\mathrm{T}} B t = 0$.

The machine can now output

$$\left[\begin{array}{c} 1 \\ \hline s \end{array}\right]^{\mathrm{T}} \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & B + Bts^{\mathrm{T}} B \end{array}\right]^* \left[\begin{array}{c} 1 \\ \hline t \end{array}\right] = \gamma^*,$$

which follows from the KAT asterate lemma. Finally, the machine can apply the Boolean axioms to each entry of

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & B + Bts^{\mathrm{T}} B \end{array}\right]$$

to produce an equivalent simple, ϵ-free transition matrix D (1 becomes the sum of all atoms). It can then output a proof of

$$\left[\begin{array}{c} 1 \\ \hline s \end{array}\right]^{\mathrm{T}} \mathrm{D}^* \left[\begin{array}{c} 1 \\ \hline t \end{array}\right] = \gamma^*.$$

The proof that the automaton constructed for a term t accepts precisely the guarded strings denoted by t is a straightforward induction.

5 Reducing the Hoare Theory of KA(T) to the Equational Theory of KA

Finally, we make the simple observation that the reductions in [2] and [5] don't significantly increase the size of the terms.

Theorem 1. *Proofs of equational implications in the Hoare Theory of KA(T) can be produced by a PSPACE transducer.*

Proof. Given an alphabet $\Sigma = \{a_1, a_2, ..., a_n\}$, let $u = a_1 + a_2 + \cdots + a_n$. In [2], it is shown that

$$s \equiv t \Leftrightarrow s + uru = t + uru$$

is a Kleene algebra congruence, therefore $(r = 0 \to p = q) \leftrightarrow (p + uru = q + uru)$. The same reduction works for KAT, as is shown in [5] - in this case u is only defined to be the sum of all of the atomic programs, not the atomic tests. The transformation from $r = 0 \to p = q$ to $p + uru = q + uru$ involves only a constant increase in size. □

6 Deterministic `while` Programs

Let P be a set of atomic programs, and B be a set of atomic tests. In [6], it is shown how to encode deterministic `while` programs as KAT terms:

$$p; q = pq$$

$$\textbf{if } b \textbf{ then } p \textbf{ else } q = bp + \overline{b}q$$

$$\textbf{if } b \textbf{ then } p = bp + \overline{b}$$

$$\textbf{while } b \textbf{ do } p = (bp)^*\overline{b}.$$

Let t be a KAT term which is an encoding of a deterministic `while` program. Let g be a guarded string over $(P \cup A_B)$. It is easy to see that the automaton (u, A, v) constructed from t in section 4 satisfies the following:

- There is only one start state s of (u, A, v) with an outgoing transition labeled by an atom x such that $\text{first}(g) = x$.
- $|\hat{\delta}_A(s, g)| \leq 1$.

Therefore, when considering the deterministic automaton (s, D, t) obtained from (u, A, v) by the standard subset construction, all states of (s, D, t) corresponding to more than one state of (u, A, v) are inaccessible. This implies that, given two KAT terms t_1 and t_2, using the above procedure to construct automata for each term and then using the procedure in [11] to generate proofs of equivalence of the automata yields proofs which are only polynomial in $|t_1| + |t_2|$.

References

[1] Berghammer, R., Möller, B. and Struth, G. (Eds.) *Relational and Kleene-Algebraic Methods in Computer Science*, May 2003.

[2] Cohen, Ernie. Hypotheses in Kleene Algebra. Technical Report TM-ARH-023814, Bellcore, 1993. http://citeseer.nj.nec.com/1688.html

[3] Cohen, E and Kozen, D. and Frederick, S. The Complexity of Kleene Algebra with Tests. *Technical Report 96-1598, Computer Science Department, Cornell University.* July 1996.

[4] Kozen, D. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Infor. and Comput*, 110(2):366-390. May 1994.

[5] Kozen, D. and Smith, Frederick. Kleene Algebra with Tests: Completeness and Decidability. *Proc. 10th Int. Workshop Computer Science Logic (CSL' 96)* 224-259. 1996.

[6] Kozen, D. Kleene Algebra with Tests. *Transactions on Programming Languages and Systems* 19:427-443. May 1997.

[7] Kozen, D. Typed Kleene Algebra. *Technical Report 98-1669, Computer Science Department, Cornell University.* March 1998.

[8] Kozen, D. On Hoare Logic and Kleene Algebra with Tests. *Trans. Computational Logic*, 1(1):60-76, July 2000.

[9] Kozen, D. and Tiuryn, Jerzy. On the Completeness of Propositional Hoare Logic. *Information Sciences*, 139:187-195, 2001.

[10] Selman, A. Completeness of Calculii for Axiomatically Defined Classes of Algebras. *Algebra Universalis*, 2:20-32, 1972.

[11] Worthington, J. Automatic Proof Generation in Kleene Algebra *Proceedings of RelMics10/AKA5*, April 2008 (to appear).